VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

TISKOVÝ SERVER PRO VYTVÁŘENÍ PDF DOKUMENTŮ

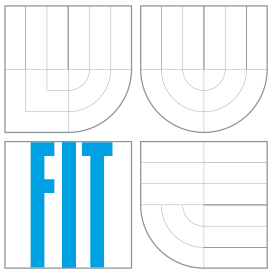
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MIROSLAV NEMEC

BRNO 2014

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

TISKOVÝ SERVER PRO VYTVÁŘENÍ PDF DOKUMENTŮ

PRINT SERVER FOR CREATING PDF DOCUMENTS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — MIROSLAV NEMEC
AUTHOR

VEDOUCÍ PRÁCE — Ing. VÍTĚZSLAV BERAN, Ph.D.
SUPERVISOR

BRNO 2014

Abstrakt

Tato práce se zaobírá tvorbou tiskového serveru pro vytváření PDF dokumentů. Výsledkem práce je řešení, které na základě vybrané a vyplněné šablony vytváří PDF dokument. Vybrání šablony, vyplnění šablony a samotná žádost o vygenerování PDF dokumentu probíhá přes webové rozhraní. Obsluhu jednotlivých požadavků zabezpečuje aplikační server. Data pro šablony jsou uloženy v databázi Caché. Tiskový server se vytváří za účelem usnadnění práce uživatelů při vytváření PDF dokumentů.

Abstract

This thesis deals with the creation and the implementation of print server for generating PDF documents. The result is the solution which generates PDF documents based on selected and filled templates. Template selection, template filling and request for generating PDF document are carried through web interface. The execution and the operation of request is handled by an application server. Data for templates are stored in the Caché database. The print server is created in order to facilitate the work of users when creating PDF documents.

Klíčová slova

tiskový server, databáze Caché, webové služby, HTML formuláře, PDF dokumenty

Keywords

print server, Caché database, web services, HTML forms, PDF documents

Citace

Miroslav Nemec: Tiskový server pro vytváření
PDF dokumentů, bakalářská práce, Brno, FIT VUT v Brně, 2014

Tiskový server pro vytváření PDF dokumentů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Vítězslava Berana, Ph.D.

.....
Miroslav Nemec
14. května 2014

Poděkování

Touto cestou bych rád poděkoval vedoucímu práce za odbornou pomoc a všem který mě podporovali při její vytváření.

Obsah

Kapitola 1

Úvod

Tato technická zpráva se zaobírá jednotlivými kroky tvorby tiskového serveru pro vytváření PDF dokumentů. Jedná se o řešení, které na základě vybrané a vyplněné šablony, vytváří PDF dokument. Vybrání šablony, vyplnění šablony a samotná žádost o vygenerování PDF dokumentu probíhá přes webové rozhraní.

V kapitole Studium(2) jsou probrány již existující řešení a technologie, které byli zapotřebí pro vytváření nového řešení nastudovat. Jedná se o studium tvorby webových služeb, webového rozhraní a PDF dokumentů. Dále se tu probírá vytváření dokumentů XML a XSL, které slouží pro generování HTML formuláře a PDF dokumentu. V neposlední řadě tu jsou probrány architektury databázových systémů a samotné databáze.

V kapitole Návrh(3) jsou probrány problémy stávajícího řešení a možnosti řešení těchto problémů. Dále tu je navrhnutí rozdělení logiky řešení, komunikace, vzhledu uživatelského rozhraní a struktury uložení dat v databázi.

Kapitola Implementace(4) se věnuje implementaci jednotlivých částí celého řešení. Jedná se o implementaci komunikace přes SOAP, klientské části, serverové části, připojení na databázi Caché, operací v Caché a samotných šablon pro vytváření HTML formulářů a PDF dokumentů.

Tiskový server se vytváří za účelem usnadnění práce uživatelů při vytváření PDF dokumentů. Celý složitý proces, při kterém se používají externí nástroje, je nahrazen jedním nástrojem.

Tiskový server pro vytváření PDF dokumentů je vytvářen a konzultován s firmou Compu Group Medical Česká republika s.r.o.

Kapitola 2

Studium

Kapitola Studium se zabývá oblastmi, které bylo pro řešení zapotřebí nastudovat. Jedná se o oblasti tvorby webových služeb, webového rozhraní a PDF dokumentů. Dále se tu probírá vytváření dokumentů XML a XSL, které slouží pro generování HTML formuláře a PDF dokumentu. V neposlední řadě tu jsou probrány architektury databázových systémů a samotné databáze. Ještě před začátkem samotného studia bylo zapotřebí prozkoumat již existující řešení.

2.1 Existující řešení

Jako zkoumané existující řešení byly použity:

- Atrium PDF Generátor
- PDF Generator
- XML Report Generator

Atrium PDF Generátor[1] je nadstavba systému Atrium, která dokáže z libovolných dat umístěných v databázi nebo z jiného zdroje automaticky vygenerovat PDF dokument v předpřipravené šabloně. Generátor umí při publikování PDF dokumentů vzít v úvahu např. aktuální den, nastavení šablony, dokáže importovat strukturovaná data do tabulky, zobrazit obrázky, aplikovat i základní typografická pravidla a umí publikovat text i do více sloupců.
Klíčové vlastnosti:

- Pohodlná tvorba PDF dokumentů bez nutnosti znalosti zlomu dokumentů v PDF.
- Volba tiskové verze (vyšší kvalita obrázků) nebo verze pro prohlížení na monitoru počítače.
- Podporované formátovací značky a objekty: kurzíva, tučné, podtržené, obrázky, odkazy, tabulky aj.
- Automatická aplikace základních typografických pravidel ve výsledném PDF dokumentu.

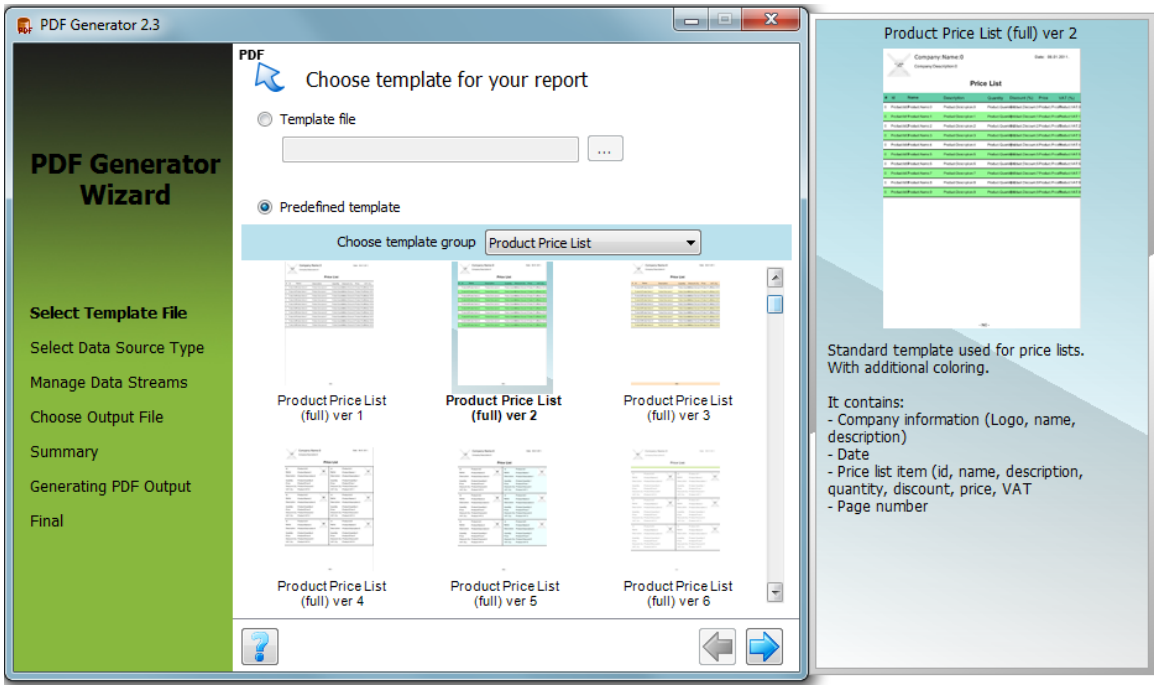
[1] `http://www.atrium-platforma.cz/cz/atrium-pdf-generator/`

- Flexibilní řízení toku textu na jednotlivých stránkách výsledného PDF dokumentu (např. dva sloupce jako v novinách).
- Generování čísel stránek, čísel vydání.
- Možnost volby z více předpřipravených šablon dle potřeby.
- Úplné propojení s redakčním systémem (sdílení obsahu).

PDF Generator[2] je aplikace sloužící pro generování PDF dokumentů, kde zdroj dat pochází z databází, excelu a popřípadě přímo z vložených dat.
Klíčové vlastnosti:

- Podpora generování dokumentů z MSSQL, MYSql, Excel, Csv, TXT.
- Možnost použití z příkazového rádku.
- V základu přináší více jak 60 předdefinovaných šablon.

Ukázka uživatelského rozhraní aplikace PDF Generator je na obrázku č. 2.1.



Obrázek 2.1: Uživatelské rozhraní aplikace PDF Generator

V levé části uživatelského rozhraní aplikace je vidět seznam kroků potřebných pro vygenerování PDF dokumentu. Zároveň se tu zobrazuje informace o tom, ve kterém kroku se proces generování PDF dokumentu aktuálně nachází. V pravé horní části je možnost vybrání vlastní šablony. Pod touto možností se nachází seznam dostupných předdefinovaných šablon. Po výběru šablony se zobrazí v pravé části aplikace panel obsahující náhled na vybranou šablonu. Mezi jednotlivými kroky vytváření PDF dokumentu se přepíná pomocí dvou tlačítek umístěných v pravém dolním rohu. K vygenerování PDF dokumentu se

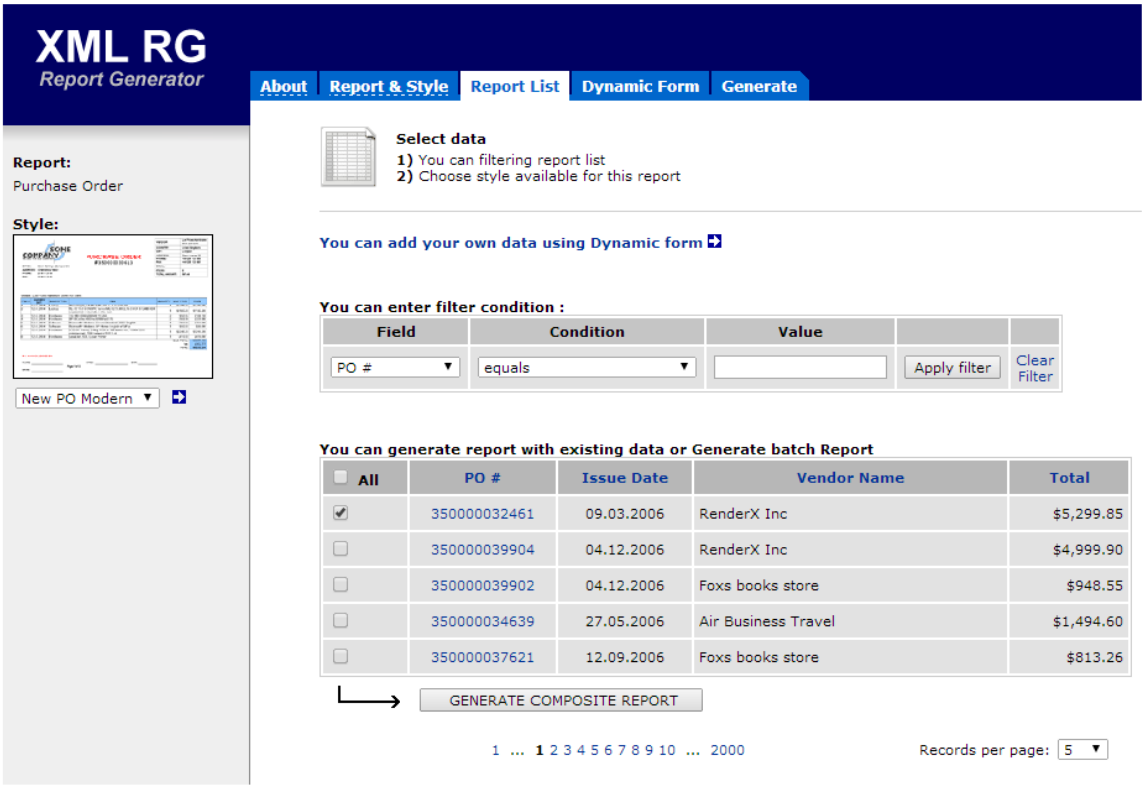
[2] `http://www.axiomcoders.com/products/applications/pdf-generator`

přichází až po vyplnění všech potřebných informací jako je například zvolení zdroje dat pro šablonu.

XML Report Generator[3] je nástroj, jehož účelem je integrovat data do výstupů a transformovat je do vysoce kvalitních tištěných dokumentů (např. PDF, PostScript, AFP). Systém pracuje s databázemi (DBS) libovolné struktury, tj. klasické relační, stejně jako post-relačí.
Klíčové vlastnosti:

- Může pracovat s jakýmkoliv typem databází (je nutná podpora JDBC, musí být podpora XML API).
- Výstupní formáty PDF, PostScript, XPS, PPML, SVG a HTML.

Ukázka uživatelského rozhraní XML Report Generator je na obrázku č. 2.2.



Obrázek 2.2: Uživatelské rozhraní XML Report Generator

V levé části uživatelského rozhraní se nachází combobox s možností výběru stylu šablony. V pravé částí se pak nachází filtr a seznam možných dat pro zvolený styl šablony. V spodní části se nachází tlačítko pro samotné generování. Po stisknutí tlačítka pro generování se systém dotazuje na typ požadovaného dokumentu (PDF, AFP, PostScript) a typ požadovaného formátu stran (A4,A3).

[3] `http://www.renderx.com/tools/xmlrg.html`

2.2 Webové služby - SOAP

Studium webových služeb bylo zaměřeno na oblast SOAP (Simple Object Access Protocol). SOAP je protokol, který komunikuje pomocí výměny správ. Správy jsou ve formátu XML. Postupy pro použití SOAP byli nastudovány z dokumentů popisujících jednotlivé části SOAP [8].

Jako aplikační vrstvu pro protokol SOAP je možné použít HTTP i SMPT. Správy se posílají přes síť hlavně pomocí HTTP, protože se více používá a je v podstatě základem pro dnešní internetovou infrastrukturu. Formát SOAP tvoří základní vrstvu komunikace mezi webovými službami a poskytuje prostředí pro tvorbu složitější komunikace.
Existuje několik různých druhů šablon pro komunikaci na protokolu SOAP:

- RPC (Remote procedure call)
- Document

RPC musí přesně dodržovat stanovený formát komunikace SOAP. Jedná se o komunikaci typu klient server. RPC je založeno na komunikaci, při které se volají vzdálené procedury. Klient tudíž jenom zobrazuje výsledky. Výhodu je možnost vypůjčit si výpočetní výkon vzdálených serverů. Nevýhodou je nutnost internetového připojení. Další nevýhodou je nemožnost použít globální proměnné a ukazatele jako parametre vzdálených procedur. U klienta se generuje balíček, který obsahuje parametry a identifikátor procedury. Pro zabalení všech potřebných informací klient odešle balíček serveru. Server si balíček rozbalí a najde odpovídající proceduru. Proceduře předá parametry a spustí ji. Po vykonaní procedury server zabalí výsledek do balíčku a zašle ho klientovy. Klient si přečte výsledek a předá ho proceduře, která si o něj žádala.

Document je v podstatě obdoba RPC s tím rozdílem že nemusí přesně dodržovat stanovený formát komunikace SOAP.

Pro popis komunikace SOAP je používáno WSDL (Web Services Description Language). WSDL popisuje jaké konkrétní funkce webová služba poskytuje. Dále definuje jak se dotazovat na konkrétní funkce webové služby. Jednoduše řečeno, WSDL popisuje veřejné rozhraní pro webovou službu.

2.3 Tvorba webového uživatelského rozhraní

Pro tvorbu webového uživatelského rozhraní se používají různé technologie.

Aby bylo možné zadefinovat obsah webové stránky je zapotřebí HTML (HyperText Markup Language)[4]. HTML je značkový jazyk určený na vytváření webových stránek.

Webové stránky musí zaujmout. Proto je jich vhodné kombinovat s CSS styly (Cascading Style Sheets)[5]. Jedná se o kolekci metod pro grafickou úpravu webových stránek. Díky ním získává webová stránka příjemný vzhled.

Aby bylo možné službu používat, je zapotřebí provádět různé operace. Jejich volání je možné díky JavaScriptu[6]. JavaScript je skriptovací programovací jazyk. Nejčastěji se používá jako interpretovaný programovací jazyk vkládaný přímo do HTML kódu stránky. Definuje například chování, které se bude provádět po stisknutí tlačítka.

[4] `http://www.w3schools.com/html/`
[5] `http://www.w3schools.com/css/`
[6] `http://www.w3schools.com/js/`

Pro přehlednost je vhodné nepsat CSS styly a JavaScripty přímo do kódu HTML. Kód by se tak stával nepřehledný. Proto je vhodné CSS styly a JavaScripty zapisovat do samostatných souborů. U CSS stylů to může být například style.css a u JavaScriptů to může být například scripts.js.

jQuery[7] je JavaScriptová knihovna zjednodušující interakci JavaScriptu s HTML. Díky ní lze například vytvořit hezkou animaci, která znázorňuje prováděnou operaci.

2.4 Tvorba PDF dokumentů

Na začátku studia bylo zapotřebí získat základní informace o PDF dokumentech[8] a možných způsobech jejich tvorby. Pro vytváření PDF dokumentů se používají různé knihovny[9]. Každý PDF dokument musí vycházet z některé verze PDF [7]. Záleží na požadavcích na výsledný PDF dokument. Hlavní funkce jednotlivých verzí PDF jsou popsány v tabulce č. 2.1.

Verze PDF	Popis funkcí
1.0	výměna dokumentů mezi platformami Windows, UNIX, Mac
1.1	přidána možnost použití šifrování a použití hesel
1.2	přidány interaktivní prvky na stránce, zvuky, unicode, komprese, podpora CMYK a přímé barvy, color managemet, atd.
1.3	přidány ICC profily, PostScript Level 3, podpora TrimBox, BleedBox, ArtBox, zvětšení max. formátu, atd.
1.4	přidány průhlednosti, XML, atd.
1.5	přidán integrovaný Preflight, atd.
1.6	přidána podpora OpenType fonts, atd.
1.7	stanoven standard ISO 32000-1:2008, od této doby pokračuje PDF jen ve verzi 1.7

Tabulka 2.1: Popis jednotlivých verzí PDF [7]

Další důležitou vlastností při vytváření PDF dokumentu je dodržovaní normy. Jedná se o normu RFC 3778[10].
PDF dokument muže být vytvářen různými způsoby:

- Použití nástrojů, které z jiných formátů (například HTML) přímo generují PDF dokumenty.
- Přímé psaní kódu PDF dokumentu.
- Generování PDF dokumentu pomocí dokumentů XML a XSL.

U nástrojů, které z jiných formátů přímo generují PDF dokumenty, se nemusíme starat o to jakým způsobem se samotný PDF dokument generuje. Nevýhodou je, že takovéto nástroje nejsou zdarma pro komerční použití.

[7] http://www.w3schools.com/Jquery/
[8] http://www.kiv.zcu.cz/ĥerout/html_sbo/pdf/toc.htm
[9] http://java-source.net/open-source/pdf-libraries
[10] http://tools.ietf.org/html/rfc3778

Přímé psaní kódu PDF dukumentu je také možné řešení. Je možné dosáhnout výsledků, kterých by se nedalo dosáhnout generováním. Nicméně jedná se asi o nejtěžší řešení.

Generování PDF dokumentu pomocí dokumentů XML a XSL je jedna ze zajímavějších cest. XML dokument obsahuje data a XSL dokument definuje vzhled výsledného PDF dokumentu.

XML dokument obsahuje například:

- ID položek.
- Jména položek.
- Výchozí hodnoty jednotlivých položek.

Existují různé typy zápisu XSL dokumentů. V rámci generování PDF dokumentů se budeme zabývat zápisem pro FOP (Formatting Objects Processor) [1]. FOP lze použít na různé účely. Asi nejběžnějším účelem je vytváření PDF dokumentů.
Základní postup pro vytváření PDF dokumentu je následující seznam kroků:

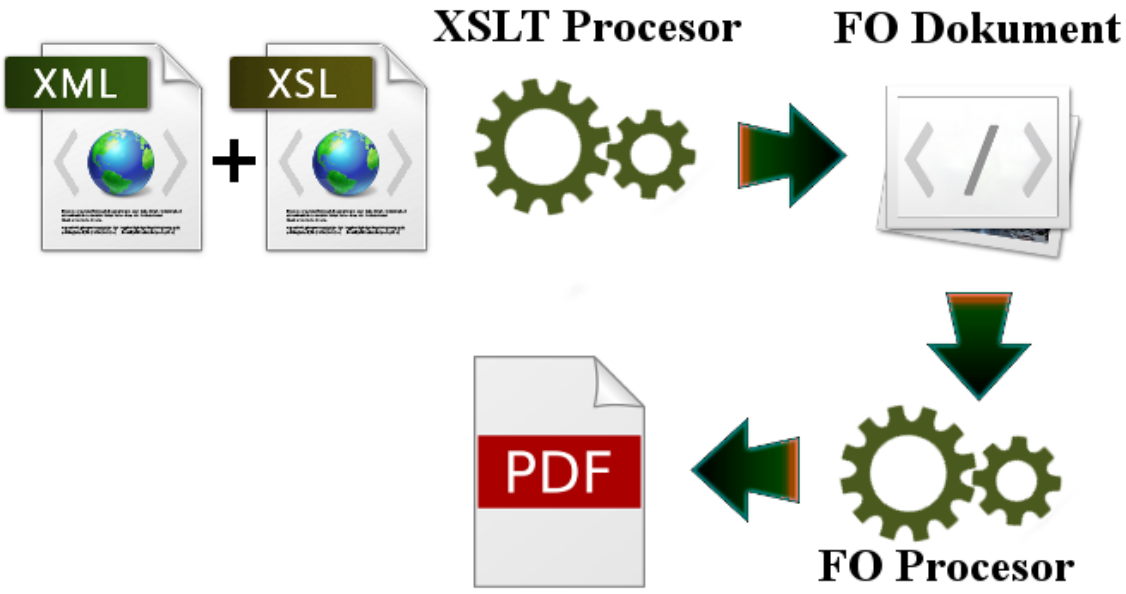
- Vytvoření dokumentu XSL-FO.
- Převedení dokumentu XSL-FO do PDF dokumentu.

XSL-FO dokument lze vytvořit buďto přímo nebo pomocí jiného jazyka. Jazyka, který je pomocí XSLT [9] předpisu převeden na XSL-FO. Zde se budeme zabývat druhou variantou. Jazyků, z kterých je možno pomocí XSLT předpisu vytvořit XSL-FO dokument není málo. Jedná se například o:

- XML
- XHTML
- TEI
- DocBook

Vytváření XSL-FO pak probíhá pomocí XSLT procesoru jehož vstupem je XSLT předpis a například XML dokument. Výsledný XSL-FO dokument je pak možné pomocí aplikace označované jako FO procesor převést na PDF dokument.

Pro zjednodušení je celý proces vytváření PDF dokumentu zobrazen na obrázku č. 2.3.



Obrázek 2.3: Převod dokumentů XML a XSL na PDF dokument [1]

Samotný XSLT předpis nepředepisuje přesný vzhled textu. Popisuje jak je rozmístěný jednotlivý obsah a vlastnosti stránek. Díky tomu může FO procesor odvodit kde bude text konkrétně umístěn. Je zapotřebí si dávat pozor na možnost rozdílných výstupů FO procesoru. Je to způsobeno různými verzi FO procesoru. Proto je vhodné si vybrat jeden vyhovující.

2.5 Tvorba HTML formulářů

HTML formulář je jedno z řešení, které umožňuje uživateli vyplnit určité údaje na webové stránce. Může se tak dít pomocí input boxů, radiobuttonů, checkboxů atd.
HTML formulář lze vytvořit dvěma způsoby.

- Napevno napsaný kód v HTML.
- Generováním kódu HTML.

Budu se tu zejména zabývat generováním HTML formulářů pomocí dokumentů XML a XSL. Je to podobné řešení jak tomu bylo v předchozí kapitole o tvorbě PDF dokumentů, ne však stejné. Zápis XSL [10] dokumentu pro HTML formulář je odlišný jak zápis XSL dokumentu pro PDF dokument.
Postup při vytvoření generovaného HTML formuláře lze shrnout do následujících bodů:

- Vytvoření XML dokumentu, který nese obsahovou část.
- Vytvoření XSL dokumentu, který obsahuje vzorec pro transformaci.
- Vytvoření HTML formuláře pomocí XSLT procesoru.

Pro zjednodušení je celý proces vytváření HTML formuláře zobrazen na obrázku č. 2.4.

Obrázek 2.4: Převod dokumentů XML a XSL na HTML formulář

2.6 Architektury databázových systémů

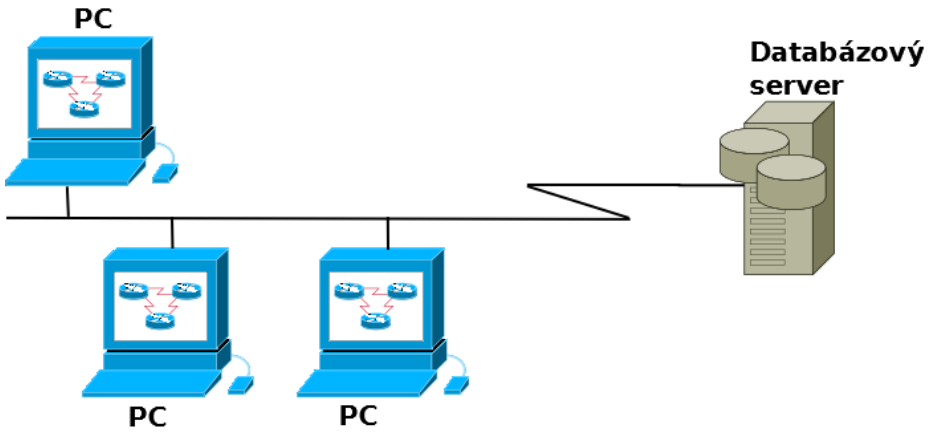
Mezi nejpoužívanější architektury databázových systémů patří:

- Dvouvrstvá architektura.
- Vícevrstvá architektura.

Dvouvrstvé architektury rozdělujeme do dvou typů:

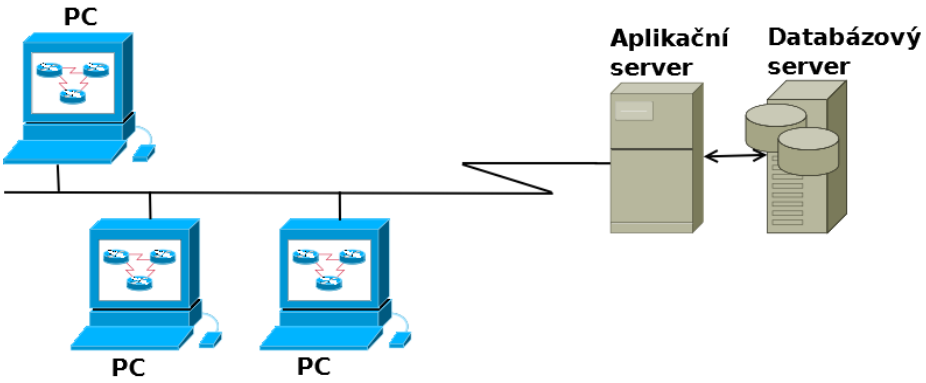
- Architektura s výkonem soustředěným u klienta.
- Architektura s výkonem soustředěným na serveru.

U architektury s výkonem soustředěným u klienta se aplikační služby zpracovávají na straně klienta. Tím mezi klientem a serverem vzniká velký počet datových přenosů. Z toho vyplývá potřeba velké přenosové kapacity. Naproti tomu architektura s výkonem soustředěným na serveru obsahuje menší počet datových přenosů. Je to dáno tím že se aplikační a datové služby zpracovávají na straně serveru. Klient pak dostává pouze požadované informace. Dvouvrstvá architektura je vidět na obrázku č. 2.5.



Obrázek 2.5: Dvouvrstvá architektura [12]

U vícevrstvé architektury klient pracuje pouze s uživatelským rozhraním. Datové a aplikační služby jsou od sebe odděleny do samostatných logických částí. Tyto části mohou být umístěny buď na stejném serveru, nebo na dvou různých serverech. Druhá varianta umožňuje získat vyšší úroveň stability. Vícevrstvá architektura je vidět na obrázku č. 2.6.



Obrázek 2.6: Vícevrstvá architektura [12]

2.7 Databázové technologie

Databázové technologie zahrnují různé typy databází. Cílem této kapitole je seznámit se s jejich základními rysy.
Mezi nejpoužívanější typy databází patří:

- Relační databáze.
- Postrelační databáze.

Mezi základní rysy relačních databází patří [12] [11]:

- Data jsou v databázi na logické úrovni organizovaná do tabulek.
- Operace nad tabulkami vytvářejí opět nové tabulky.
- Vztahy mezi řádky tabulek jsou vyjádřeny pomocí cizích a kandidátních klíčů.
- Omezená množina datových typů hodnot (skalární/atomické).

Postrelační databáze dále rozdělujeme například na:

- Objektově orientované.
- Objektově relační.

Mezi základní rysy objektově orientované databáze patří [11]:

- Umožňují modelování a vytváření perzistentních dat jako objektů.
- Jednoznačný identifikátor objektu pro každý perzistentní objekt
- Podpora abstraktních datových typů, zapouzdření a polymorfismu.

- Atributy objektů mohou být jiné objekty - složité typy.

Objektově relační databáze si klade za cíl spojit výhody relačního a objektového modelu. Mezi základní rysy objektově relační databáze patří [11]:

- Snaha o obohacení tabulek o objektovou orientaci.
- Z hlediska datové struktury jde o obecnější relace.
- Perzistentní data jsou stále v tabulkách, ale hodnoty mohou mít bohatší strukturu definovanou jako abstraktní datový typ.
- Abstraktní datový typ zapouzdřuje data a operace.
- Je zavedena obdoba identifikátoru objektu, která umožňuje vytvářet nový typ vazeb mezi tabulkami.

Studium databázových technologií bylo zaměřeno na studium postrelační databáze Caché [4]. Firma InterSystems jejíž nosným produktem je právě Caché byla založena v roce 1978. Caché je velmi vyspělý nástroj pro tvorbu komplexních aplikací založených na práci s persistentními daty [2].

Na rozdíl od klasických relačních i čistě objektových platforem je Caché díky své architektuře uložení dat schopna pracovat s daty jak z pohledu objektového, tak i relačního. Základem modelování komponent aplikací jsou v Caché objekty. Objekty jsou organizovány do tříd s vlastnostmi a metodami. Definice tříd spolu s ostatními daty jsou uloženy v jednotném úložišti zvaném Caché Class Dictionary. Caché Class Dictionary tvoří databázi, ke které můžeme objektově přistupovat. Zkompilováním definice třídy dojde k vytvoření dvou různých, navzájem synchronizovaných sad kódu, které zajistí optimální přístup k datům jak pomocí objektového, tak relačního přístupu. Fyzické úložiště dat se v Caché nazývá globál. Jedná se o vícerozměrné řídké pole.

Caché je nástupcem předchozí databázové technologie Mumps. Studium tím pádem začínalo studiem příručky pro Mumps [6].

InterSystems poskytuje příručky pro připojení do databáze Caché pomocí nejpoužívanějších programovacích jazyků. Například pro programovací jazyk Java [3].

Kapitola 3

Návrh

Kapitola návrh se zabývá návrhem jednotlivých částí řešících tiskový server pro generování PDF dokumentů. Jedná se o navrhnutí rozdělení logiky řešení, komunikace, vzhledu uživatelského rozhraní a struktury uložení dat v databázi. Aby bylo však možné začat navrhovat jednotlivé části, je zapotřebí zadefinovat problém a navrhnout možné řešení tohoto problému.

3.1 Definice problému

Firma, která tvoří IS pro nemocnice, potřebuje škálovatelné řešení pro snadné generování PDF dokumentů z vyplněných formulářů. Současné řešení není vyhovující a proto je jej třeba změnit.

Při stávájícím řešení si uživatel stáhne na svůj disk šablonu, v externím nástroji tuto šablonu vyplní a pomocí dalšího externího nástroje generuje PDF dokument. Provedení výše zmíněných úkonů zabírá zbytečně mnoho času. Času kterého v dnešním rychleném světě není nikdy dostatek. Kromě zbytečného spotřebovaného času tady vzniká množství problémů. Problémů jako je aktuálnost verze šablony. Uživatelé nekontrolují aktuální verzi šablony. Jednoduše používají tu kterou mají uloženou na svém disku. Při aktualizaci externí aplikace může nastat problém konzistence šablony/PDF dokumentu. Další problém je závislost použitých aplikací na operačním systému.

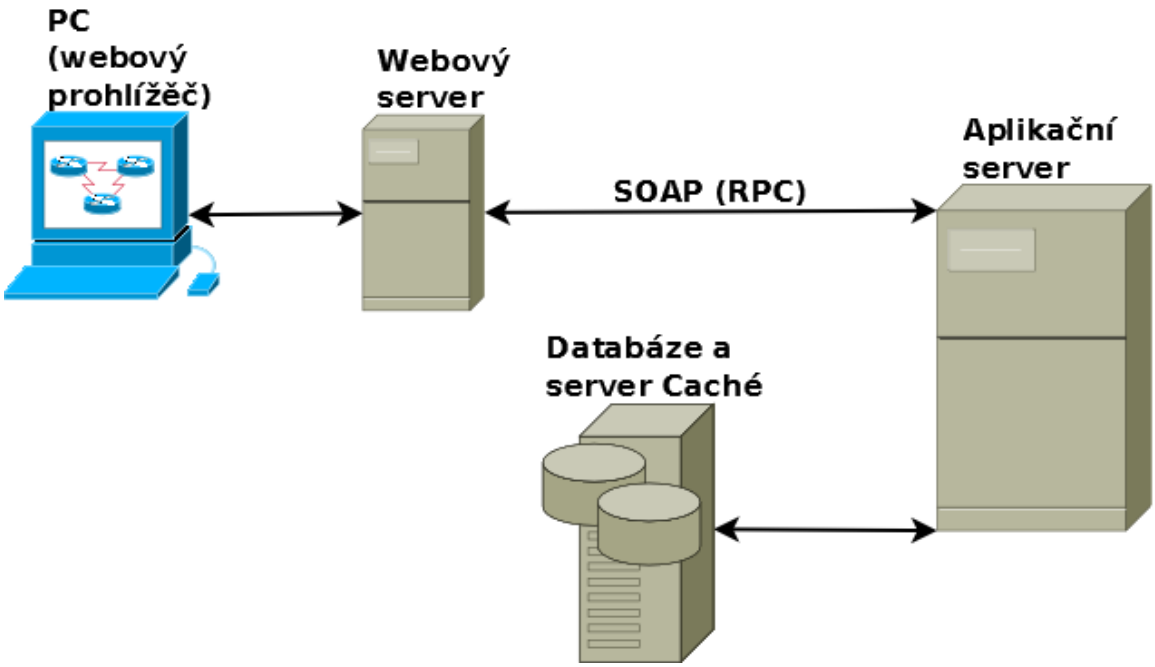
3.2 Návrh řešení problému

Kvůli již zmíněným problémům je zapotřebí vytvořit systém, který tyto problémy efektivně vyřeší. Na začátku je otázka jak zefektivnit každodenní práci uživatelů. Lze to zabezpečit pomocí webu, který zapouzdřuje webové služby. Takovýto web pak nahradí aplikace pro stahování šablon, editaci šablon a generování PDF dokumentů. Jinak řečeno vše v jednom. Pro používání webových služeb je vhodné použít technologie SOAP. Jak již bylo zmíněno v kapitole studium, existuje několik různých druhů šablon pro komunikaci na protokolu SOAP. Pro tiskový server je vhodné použít šablonu RPC. Veškerou výpočetní práci bude zastávat server. Použitím webových služeb rovněž řeší problém konzistence šablony/PDF dokumentu. Výsledný PDF dokument bude vždy z dané webové služby strukturově stejný. Problém aktuálnosti verze šablony lze řešit použitím verzování na straně databáze a serveru, který poskytuje dané webové služby. To znamená že by při každé návštěvě služby uživatel obdržel šablonu s nejnovějším číslem verze. Jelikož jsou weby obsahující webové

služby platformově nezávislé, otevře se cesta i pro ty, které omezuje operační systém. Jako architektura databázových systému byla zvolena architektura vícevrstvá.
Z pohledu požadavků na funkcionalitu lze cílové řešení zobecnit do následovných bodů:

- Uživatel má možnost vybrat/plnit šablonu pro PDF dokument.
- Šablona se načítá/ukládá z/do databáze Caché.
- Server na základě zvolené a vyplněné šablony pro PDF dokument generuje PDF dokument.

Výše navrhované řešení lze jako celek znázornit na obrázku č. 3.1.



Obrázek 3.1: Návrh řešení

Logika aplikace je rozdělena na menší oddělené části.
Rozdělení logiky proběhlo nad následujícími částmi:

- Webový server (klientská část).
- Aplikační server (serverová část).
- Databáze a server Caché.

Webový server bude obsahovat pouze malou část logiky.
Jedná se o:

- Definici vzhledu webu.
- Ovládací prvky webu.
- Volání webových služeb.
- Přijímání dat.

Webový server bude moct data přijímat z více zdrojů.
Z aplikačního serveru:

- Ve formě seznamu dostupných šablon.
- Ve formě URL adresy HTML formuláře představujícího vybranou šablonu.
- Ve formě samotného PDF dokumentu.

Z uživatelského vstupu:

- Ve formě dat z vyplněného HTML formuláře představujícího vybranou šablonu.

Aplikační server bude obsahovat největší část logiky.
Jedná se o:

- Webové služby.
- Přístup k databázi Caché.

Konkrétní požadavky, které budou přicházet na aplikační server:

- Žádost o seznam dostupných šablon.
- Žádost o vygenerovaní HTML formuláře představujícího vybranou šablonu.
- Žádost o vygenerovaní PDF dokumentu.

Konkrétní odpovědi které budou odcházet z aplikačního serveru:

- Seznam dostupných šablon.
- URL vygenerovaného HTML formuláře představujícího vybranou šablonu.
- Vygenerovaný PDF dokument.

Server Caché bude obsahovat třídy, které zpřístupňují, popřípadě připravují data pro server. Bude se jednat se o data, které nesou informace o šablonách pro PDF dokumenty a samotný obsah pro ně. Tyto data budou umístěny v databázi Caché.

Jako programovací jazyk na straně webového serveru a aplikačního serveru bude použit jazyk JAVA. Jeho hlavní výhodou je platformová nezávislost. Při tvorbě bude využíváno vlastností objektově orientovaného jazyka. Další výhodou je přímá podpora připojení na databázi Caché. Jako vývojové prostředí bude použito vývojové prostředí NetBeans. NetBeams obsahuje řadu ukázkových příkladu [5], které usnadní práci na začátku. Existuje celá řada programovacích jazyků, které by se dali ještě použít. Jedná se například o jazyk Delphi, který má taktéž přímou podporu připojení na databázi Caché.

Jako webový server bude použit GlassFish[1], který je již součástí instalace NetBeans. Dále by se mohl použít například webový server Apache Tomcat[2].

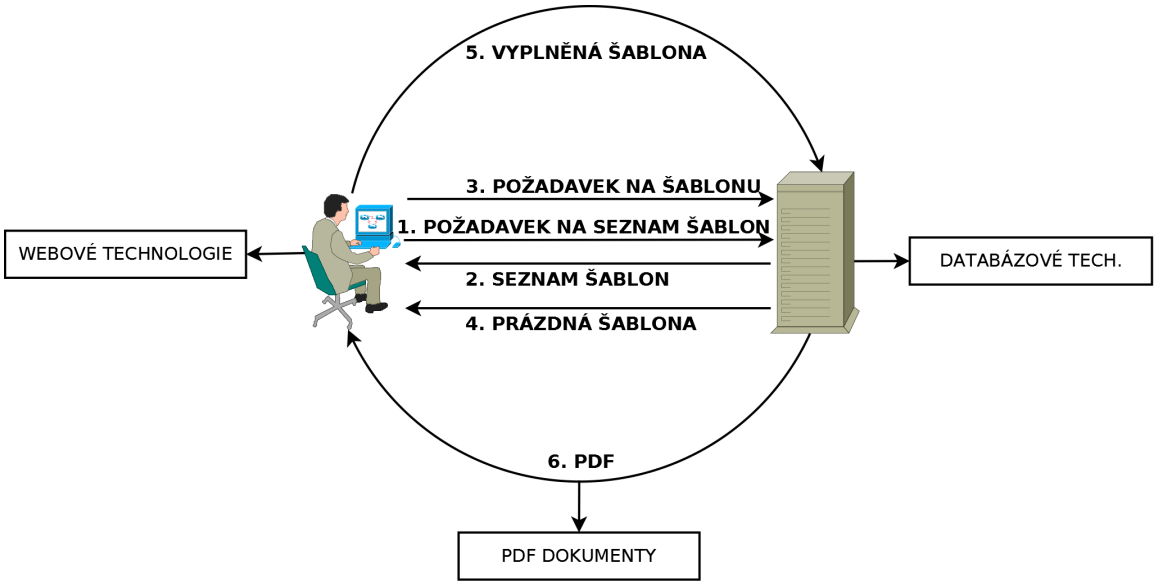
Pro definice šablony bude zapotřebí mít dvě definice šablon (XSL dokumenty). Jedna definice bude sloužit pro vygenerování HTML formuláře. Formuláře, který bude uživatel vyplňovat. Druhá definice bude sloužit pro samotné generování PDF dokumentu. V obou případech je zapotřebí stejného XML dokumentu, který nese informace o datech.

Podrobnější návrh vybraných částí řešení bude probrán v následujících podkapitolách. V dalších částech textu se bude webový server nazývat jako klientská část neboli klient a aplikační server se bude nazývat jako serverová část neboli server.

[1] `https://glassfish.java.net/documentation.html`
[2] `http://tomcat.apache.org/`

3.3 Návrh komunikace

Pro komunikaci byla zvolena komunikace typu klient-server. Celkový přehled komunikace mezi jednotlivými částmi je zobrazen na obrázku č. 3.2.



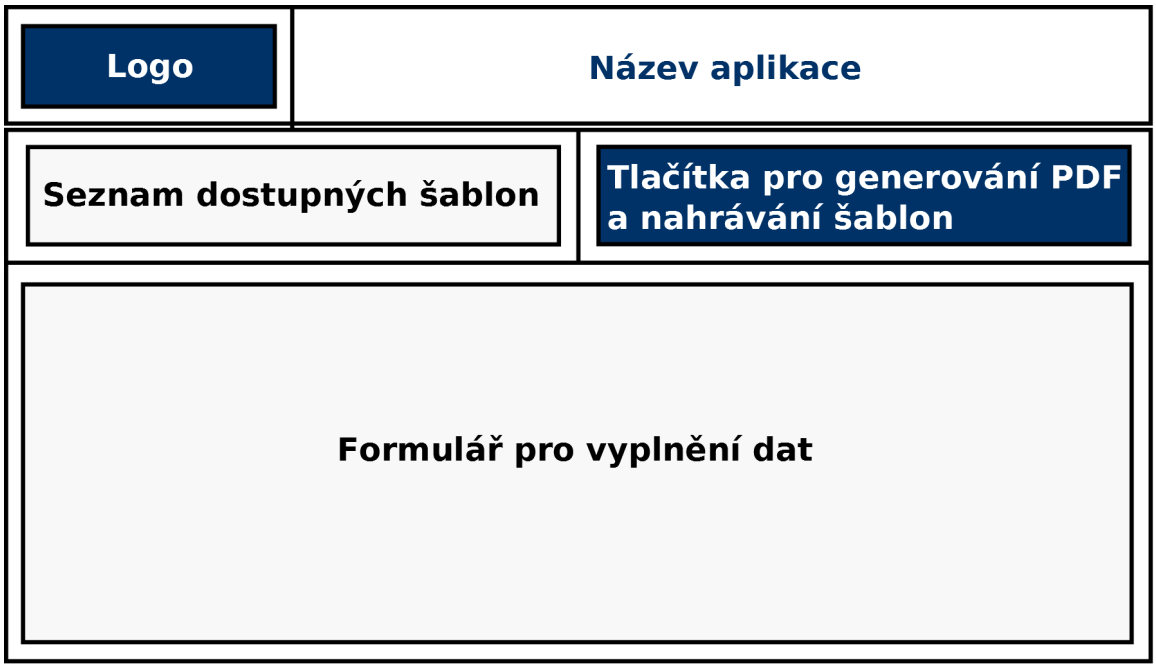
Obrázek 3.2: Návrh komunikace

Průběh komunikace lze popsat v následujících bodech:

- Klient zašle požadavek na server o seznam dostupných šablon pro PDF dokument.
- Server zašle klientovy seznam dostupných šablon pro PDF dokument.
- Klient zašle požadavek na server o vybranou šablonu pro PDF dokument.
- Server zašle klientovy URL adresu HTML formuláře představujícího vybranou šablonu.
- Klient vyplní HTML formulář představující vybranou šablonu.
- Klient zašle serveru data z vyplněného HTML formuláře spolu s názvem aktuálně vybrané šablony.
- Server vygeneruje PDF dokument, který následně zašle klientovy jako přílohu.

3.4 Uživatelské rozhraní

Grafické uživatelské rozhraní, viz. obrázek č. 3.3, je navrženo tak, aby bylo co nejjednodušší a nejpřehlednější.



Obrázek 3.3: Návrh uživatelského rozhraní

Na samotném vrchu obrazovky se budou nacházet identifikační údaje jako je logo firmy a název služby. Pod nimi se budou nacházet ovládací prvky služby. Na levé straně bude combobox obsahující seznam dostupných šablon. Na pravé straně budou dvě tlačítka pro nahrávání definici šablony a pro generování PDF dokumentu. Ve středě obrazovky bude hlavní oblast. Jedná se o oblast, ve které bude uživatel vyplňovat HTML formulář představující vybranou šablonu.

V budoucnu je v plánu zavést možnost autorizace. Poté bude vytvořeno množství nových funkcí. Tyto funkce bude také třeba zakomponovat do uživatelského rozhraní. Asi nejlepší možností je umístit skrývatelný panel na pravou stranu uživatelského rozhraní. Po přihlášení bude na pravé straně uživatelského rozhraní ikona. Po kliknutí na tuto ikonu se zobrazí/skryje výše zmíněný panel.

3.5 Návrh struktury uložení dat

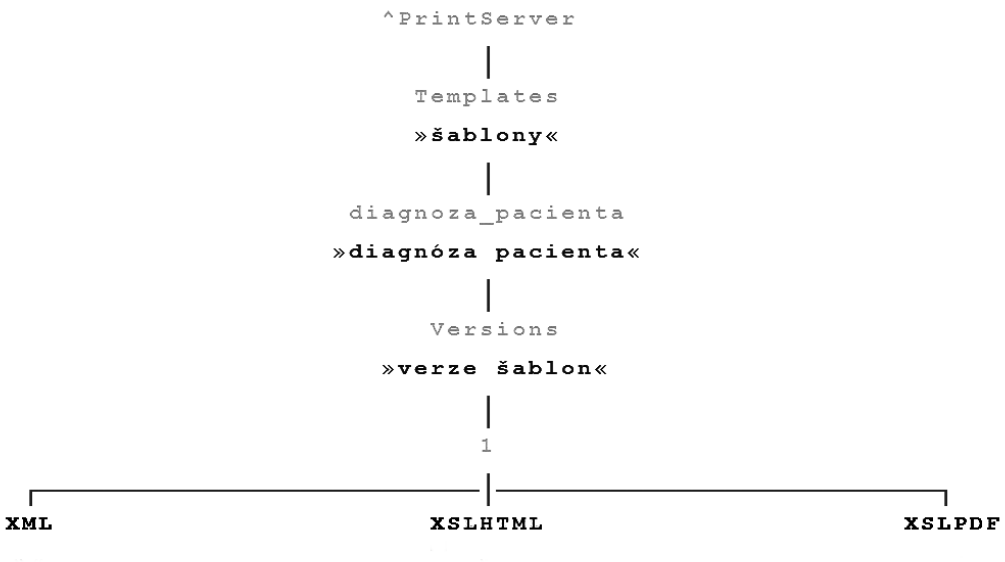
Na začátku bylo zapotřebí navrhnout strukturu uložení dat šablon v databázi Caché.

Jak již bylo zmíněno v podkapitole č. 2.7, fyzické úložiště dat se v Caché nazývá globál, což je vícerozměrné řídké pole. Na obrázku č. 3.4 je vidět struktura uložení dat šablon v databázi Caché.

```
^PrintServer("Templates")
^PrintServer("Templates","diagnoza_pacienta","Versions",1,"XML")
^PrintServer("Templates","diagnoza_pacienta","Versions",1,"XSLHTML")
^PrintServer("Templates","diagnoza_pacienta","Versions",1,"XSLPDF")
^PrintServer("Templates","protokol_darce_krve","Versions",1,"XML")
^PrintServer("Templates","protokol_darce_krve","Versions",1,"XSLHTML")
^PrintServer("Templates","protokol_darce_krve","Versions",1,"XSLPDF")
```

Obrázek 3.4: Struktura uložení dat šablon v databázi Caché

Pro jednoduchost je na obrázku č. 3.5 pomocí stromu zobrazena struktura uložení dat šablon v databázi Caché.



Obrázek 3.5: Zobrazení struktury uložení dat šablon v databázi Caché pomocí stromu

Jak je vidět na obrázcích č. 3.4 a 3.5, globál byl nazván PrintServer. Pod tímto globálem se nachází index Templates. Pod indexem Templates se nacházejí indexy s názvem jednotlivých šablon. Každý index s názvem šablony obsahuje index Versions. Pod tímto indexem se nacházejí indexy pojmenované číslem a to konkrétně číslem verze šablony. Pod indexem čísle verze šablony se nacházejí tři indexy. Jedná se o indexy XML, XSLHTML a XSLPDF. Jejich obsah definují již zmíněné dokumenty pro vytváření HTML formulářů a PDF dokumentů.

Kapitola 4

Implementace

Kapitola implementace se zabývá postupnou implementací jednotlivých částí celku. Jedná se o implementaci komunikace přes SOAP, připojení na Caché, serveru, který obsahuje webové služby, klienta, který volá webové služby, operací v Caché a webového rozhraní pro komunikaci s uživatelem. Taktéž se tady nachází implementace šablon pro HTML formulář a PDF dokument.

4.1 Komunikace přes SOAP

Na začátku jsem pro pochopení potřebných vlastností SOAP využíval ukázkové příklady. Jak již bylo řečeno v podkapitole č. 3.2, jako vývojové prostředí bylo zvoleno NetBeans. Ukázkové příklady rovněž pocházely z vývojového prostředí NetBeans. Jako ukázkový příklad jsem použil webovou aplikaci kalkulačka. Tato webová aplikace na straně klienta obdrží dvě čísla. Tyto čísla zašle serveru. Server tyto čísla sečte a zašle spět klientovy. Pomocí tohoto ukázkového příkladu bylo snadnější zjistit, jaké programové části je zapotřebí implementovat. Následně jsem vytvořil vlastní příklady. Výsledkem byli základní stavební znalosti pro projekt tiskový server pro vytváření PDF dokumentů.
Konkrétní vlastní příklady:

- Konvertování textu. Klient napíše text, server jej upraví a zašle zpět.
- Jednoduchá webová aplikace získávající informace o Caché.

Implementace - strana serveru:

- Založení projektu - Web Application.
- Vytvoření služby (WebService) - třída, která zapouzdřuje funkce, které se pak budou volat z klienta.
- Nastavení adresy serveru.
- Nastavení webové stánky serveru, která může například informovat o aktuálním stavu serveru.

Implementace - strana klient:

- Založení projektu - Web Application.

- Vytvoření třídy, která obsahuje připojení na server spolu s konkrétním požadavkem na něj.
- Vytvoření WebServlet, pod jehož jménem se bude volat předchozí třída.
- Nastavení adresy klienta.
- Nastavení webové stránky klienta, ze které se budou volat WebServlet.

Po úspěšném zprovoznění již zmíněných příkladů byl nalezen zásadní problém a to konkrétně problém pevně nastavených IP adres a portů. To způsobí problém při instalaci celého řešení. Na straně klienta je nutné mít možnost editovat adresu serveru obsahujícího volané služby. Na straně serveru je pak nutnost vědět svou vlastní IP adresu a port. Ta se použije například při potřebě zaslání odkazu na soubor umístěný na serveru klientovy. Na straně klienta byl tento problém vyřešen editací WSDL souboru. Klient si pro každou volanou službu uchovává WSDL soubor, který obsahuje údaje o dané službě. Administrátor pak při instalaci klienta může jednoduše upravit adresy a porty jednotlivých služeb. Na straně serveru byl založen konfigurační soubor, viz. příloha F, který obsahuje IP adresu a číslo portu serveru.

4.2 Připojení na Caché

Pro první pokus připojení na server Caché byla použita ukázková knihovna jar od InterSystems. Tato knihovna měla za úkol připojit se k serveru Caché a následně získat informace o serveru Caché. První pokus proběhl úspěšně. Dalším pokusem měla být vlastní knihovna, která bude vracet mnou určené data z databáze Caché. Dle příručky od InterSystems [3], je zapotřebí nejprve v Caché vytvořit třídu obsahující metody, které implementují požadované operace nad databází. Následně se pomocí speciálního příkazu vygeneruje jar knihovna. Tato knihovna pak zprostředkovává výše zmíněnou třídu a její metody. Ještě před samotným voláním metod třídy v knihovně jar, je zapotřebí vytvořit spojení na databázi Caché. Zde nastal dlouho řešený problém. Jádro pro připojení k serveru Caché bylo použito z ukázkového příkladu. Nicméně v mém příkladu nefungovalo. Nakonec byla zjištěná příčina problému. Připojení na server Caché může dle ukázkového příkladu běžet ve dvou režimech. Buďto normální připojení nebo tzv. liteconnection. Ukázkový příklad fungoval na druhé variantě. Nicméně vlastní příklady vyžadují první variantu. Celý problém se tudíž vyřešil nastavením této jedné vlastnosti. Po úspěšném zprovoznění připojení na Caché byla v projektu pro obecnost vytvořená třída ConnectToCache.
Tato třída obsahuje dvě metody:

- Connect
- ConnectWithXMLConfig

Metoda Connect se na základě zadaných parametrů připojí na Caché.
Jedná se o parametry:

- host - IP adresa serveru Caché
- port - port serveru Caché
- nameSpace - cílový name space

- username - uživatelské jméno
- password - uživatelské heslo

Metoda ConnectWithXMLConfig načte konfigurační soubor, jenž obsahuje výše zmíněné parametry, a následně zavolá s těmato parametry funkci Connect.

4.3 Serverová část

Výhodou při implementaci SOAP byla přímá podpora NetBeans. Některé části, jako jsou volání vzdálených procedur, lze automaticky generovat. Na začátku bylo zapotřebí naprogramovat třídy a jejich metody, které se budou pak následně volat přes SOAP.
Seznam metod:

- Metoda, která vrací seznam dostupných šablon (třída LoadTemplate).
- Metoda, která na základě zvolené šablony vrací vygenerovaný HTML formulář (třída LoadTemplate).
- Metoda, která na základě dat z vyplněného HTML formuláře vrací vygenerovaný PDF dokument (třída GeneratePDF).

Zde bude probrán postup implementaci každé výše zmíněné metody. Bude se tu procházet pouze část, která je implementovávaná na serveru. Části, které jsou implementovány v Caché, budou probrány v podkapitole č. 4.5.

Metoda, která vrací seznam dostupných šablon, byla pojmenována jako GetTemplateNames. Tato metoda nepotřebuje žádné vstupní parametry. Samotný požadavek o získání seznamu šablon deleguje na Caché.

Metoda, která na základě zvolené šablony vrací vygenerovaný HTML formulář, byla pojmenována jako GetTemplate. Tato metoda vyžaduje vstupní parametr a to jméno šablony. Toto jméno je získáno z klientské části. Jak již bylo zmíněno v podkapitole č. 3.2, na to aby se mohl vygenerovat HTML formulář je zapotřebí dvou dokumentů. Jedná se o dokumenty XML a XSL. Obsah těchto dokumentů se připravuje na straně Caché. Server si postupně vyžádá obsah pro jednotlivé dokumenty. Jako parametr žádosti se vždy předává jméno šablony a typ požadovaného dokumentu. Následně jsou tyto dokumenty uloženy fyzicky na disk. Je to z toho důvodu, aby se při dalším požadavku nemuseli dokumenty opětovně získávat z databáze Caché. Zde nastává však problém aktualizace šablony. Proto se před požadavkem o obsah šablon posílá požadavek o číslo aktuální verze šablony v databázi Caché. Pokud se toto číslo nebude shodovat s číslem na serveru, bude proveden požadavek o šablony. V jiném případě není zapotřebí získávat obsah jednotlivých dokumentů a tím pádem je možno přeskočit další operace. Jediná výjimka, kde nezáleží na verzi šablony, je situace, kdy server ještě nemá žádnou verzi požadované šablony.

Jak již bylo zmíněno v podkapitole č. 3.2, pro vytváření HTML formuláře je zapotřebí jiný XSL dokument jak pro vytváření PDF dokumentu. Stačilo by tudíž převzít pouze XSL dokument pro HTML formulář. Nicméně pro kompletnost se v tomto kroku získává i obsah XSL dokumentu pro PDF dokument.

Po úspěšném uložení všech potřebných dokumentů je zahájen proces generování HTML formuláře. Na vstup generátoru přichází dokumenty XML a XSL (určený pro HTML formulář). Následně je vygenerovaný obsah pro HTML formulář uložen na disk. Nakonec volaná metoda vrací žádajícímu klientovy URL adresu tohoto HTML formuláře. Toto se

děje i v případě kdy nebyla zapotřebí aktualizace obsahu jednotlivých dokumentů (HTML formulář již existoval).

Pro příklad tu uvedu adresářovou strukturu šablon uložených na serveru.
Jako příkladová šablona byla použita šablona s jménem protokol_darce_krve:

- ——— Templates
- ——— Templates ——— protokol_darce_krve
- ——— Templates ——— protokol_darce_krve ——— protokol_darce_krve_ver.txt
- ——— Templates ——— protokol_darce_krve ——— protokol_darce_krve_HTML.xsl
- ——— Templates ——— protokol_darce_krve ——— protokol_darce_krve_PDF.xsl
- ——— Templates ——— protokol_darce_krve ——— protokol_darce_krve.xml
- ——— Templates ——— protokol_darce_krve ——— protokol_darce_krve.html

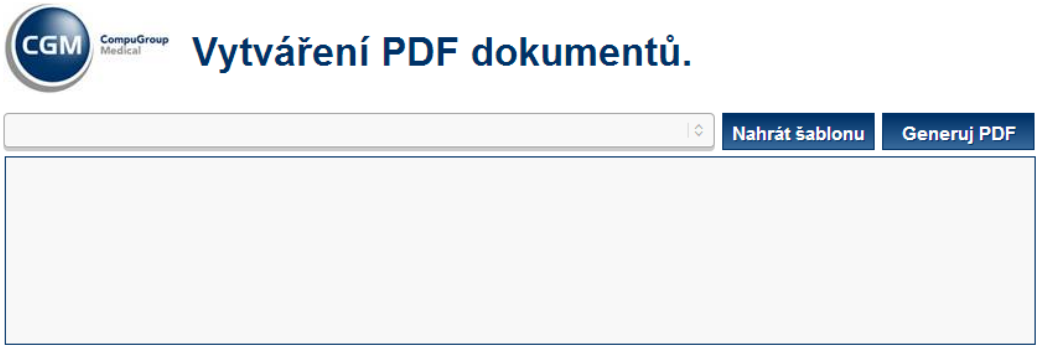
Soubor protokol_darce_krve_ver.txt obsahuje číselný údaj o verzi šablony uložené na serveru. Soubor protokol_darce_krve_HTML.xsl obsahuje XSL dokument, který se použije při tvorbě HTML formuláře. Soubor protokol_darce_krve_PDF.xsl obsahuje XSL dokument, který se použije při tvorbě PDF dokumentu. Soubor protokol_darce_krve.xml obsahuje datovou část šablony. Soubor protokol_darce_krve.html obsahuje výsledný HTML formulář, který vnikl procesem generování.

Metoda, která na základě dat z vyplněného HTML formuláře vrací vygenerovaný PDF dokument, byla pojmenována jako GeneratePDF. Tato metoda vyžaduje dva vstupní parametry. Jedná se o jméno šablony a pole dvojic obsahujících hodnoty z vyplněného HTML formuláře. Dvojice je složena z ID položky a samotné hodnoty položky. Metoda nejprve načte obsah dokumentu XML, který definuje datovou část šablony. Dále se nad XML dokumentem v paměti provede aktualizace všech hodnot identifikovaných pomocí ID. V podstatě se přenášejí informace z vstupního pole do dokumentu XML. Takto upravený dokument XML (prozatím uložen pouze v paměti) je uložen do temp adresáře. Jméno tohoto XML dokumentu je generováno na základě aktuálního data a času. Po úspěšném uložení temp XML dokumentu přichází na řadu generátor PDF dokumentů. Ten na vstup vyžaduje již zmíněný temp XML dokument a XSL dokument. Výstupem generátoru je PDF dokument. Jméno PDF dokumentu je stejné jako jméno vytvořeného XML dokumentu. Server následně vrací klientovy URL adresu PDF dokumentu.

4.4 Klientská část

Klientská část je tvořena webovým uživatelským rozhraním a částí, která předává požadavky klienta serveru.

Webové rozhraní je implementované pomocí kombinace HTML, CSS stylů, JavaScriptu a jQuery. HTML v kombinací s CSS styly definuje samotný vzhled webu. Dále je pomocí HTML definován vzhled webového formuláře, který představuje vybranou šablonu. JavaScript se používá pro volání vzdálených procedur (webových služeb serveru). jQuery se používá pro sesbíraní dat, které uživatel vložil do formuláře. Výsledné webové uživatelské rozhraní je vidět na obrázku č. 4.1.



Obrázek 4.1: Webové uživatelské rozhraní

V následující části textu bude zmiňovaná klientská část myšlena jako část, která předává požadavky klienta na server.

Klientská část je v podstatě most mezi webovým rozhraním a serverem. Přijímá požadavky od klienta a předává je serveru. Následně zpracovanou odpověď serveru vrací webovému rozhraní.
Klientská část obsahuje následující metody:

- Metoda, která předává požadavek o seznam dostupných šablon (třída Client_LoadTemplateNames).
- Metoda, která na základě zvolené šablony předává požadavek o HTML formulář (šablonu) (třída Client_LoadTemplate).
- Metoda, která na základě dat z vyplněného HTML formuláře předává požadavek na PDF dokument (třída Client_GeneratePDF).

Samotné metody neobsahují žádnou rozšiřující logiku. Jak již bylo zmíněno, je to pouze most k metodám již zmíněným v podkapitole č. 4.3

4.5 Operace v Caché

V Caché jsou implementované třídy, jejichž metody se vyvolávají prostřednictvím serveru. Základní třída obsahuje několik metod.
Jedná se o:

- Metodu, která prochází databázi a vrací seznam dostupných šablon.
- Metodu, která na základě jména šablony vrací aktuální číslo verze šablony umístěné v databázi.
- Metodu, která na základě jména šablony vrací definici šablony. Jedná se o obsah pro dokumenty XML, XSL pro HTML formulář a XSL pro PDF dokument.

Všechny výše zmíněné metody pracuji nad strukturou, která byla navrhnuta v podkapitole č. 3.5, viz. obrázek č. 3.4. Pro jednoduchost tady budu používat stromovou strukturu viz. obrázek č. 3.5, který zjednodušeně reprezentuje skutečné uložení struktury dat v Caché.

Metoda, která prochází databázi a vrací seznam dostupných šablon, nevyžaduje žádný vstupný parametr. Metoda prochází jednu úroveň podstromu PrintServer(Templates). Při procházení tohoto podstromu si metoda ukládá jméno každého procházeného uzlu. Jedná se o jméno šablony. Jednotlivé jména šablon jsou spolu s oddělovačem ukládána za sebou do lokální proměnné. Po projetí celého podstromu metoda vrací již zmíněnou lokální proměnnou plnou jmen šablon.

Metoda, která na základě jména šablony vrací aktuální číslo verze šablony umístěné v databázi, vyžaduje jeden vstupní parametr a to jméno šablony. Následně tato metoda načte odzadu jeden uzel v podstromu PrintServer(Templates, jméno šablony, Versions). Touto operací metoda získá poslední číslo verze šablony.

Metoda, která na základě jména šablony vrací definici šablony, vyžaduje jako vstupní parametr jméno šablony a typ požadovaného dokumentu. Na základě typu požadovaného dokumentu metoda načte obsah jednoho z následujících uzlů podstromu
PrintServer(Templates, jméno šablony, Versions, poslední verze šablony):

- XML
- XSLHTML
- XSLPDF

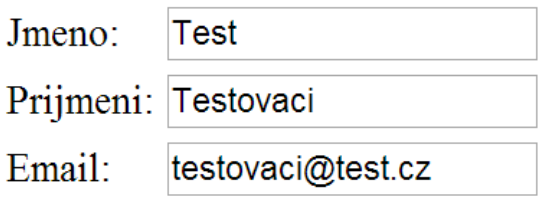
Následně metoda vrací tento obsah.

Dále bylo zapotřebí vytvořit rutinu, která inicializuje struktury pro uložení šablon v Caché. Tato rutina vytvoří strukturu již zmíněnou v podkapitole č. 3.5. Spouští se v terminálu zadáním příkazu `d CreateTreeTemplates^Templates`. Po vytvoření struktury pro uložení šablon v Caché bylo zapotřebí vytvořit rutinu, která nahraje konkrétní šablony. Spouští se v terminálu zadáním příkazu
`d FillTemplate^Templates(templateName,version)`. Tato rutina má na rozdíl od té předchozí dva parametry. Jméno nahrávané šablony a verzi nahrávané šablony. Výsledkem je podobná struktura jako je ukázková struktura viz obrázek č. 3.4.

4.6 Šablona pro HTML formulář a PDF dokument

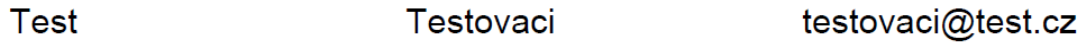
Před začátkem vytvářeních reálných šablon bylo vhodné začít s vytvářením jednoduché šablony. Úplně na začátku bylo zapotřebí definovat zdrojové data šablony. Jak již bylo zmíněno v podkapitole č. 3.2, jedná se o XML dokument. Ukázka XML dokumentu je uvedena v příloze A. V XML dokumentu bylo zapotřebí pomocí tagu `<!ATTLIST * id ID #IMPLIED>` definovat jednotlivé ID XML dokumentu. Pro samotné generování HTML formuláře to není zapotřebí. Zapotřebí je to až ve chvíli, kdy se bude generovat PDF dokument. Generátor, který vytváří PDF dokument, nepozná ID a proto je jej nutno tímto způsobem zadefinovat.

Dále bylo zapotřebí definovat vzhled šablony pro HTML formulář. Jak již bylo zmíněno v podkapitole č. 3.2, jedná se o XSL dokument. Ukázka XSL dokumentu pro HTML formulář je uvedena v příloze B. Nejdůležitější části XSL dokumentu pro HTML formulář jsou části, ve kterých se definuje cesta k hodnotám jednotlivých tagů XML dokumentu. Jedná se například o tag `<xsl:value-of select=‘‘cesta_v_xml/value‘‘/>`. V případě kdy potřebujeme hodnotu z definice tagu, například caption, použijeme obdobný tag s tím rozdílem, že se použije znak @. Například `<xsl:value-of select=‘‘cesta_v_xml/@caption‘‘/>`. V neposlední řadě je zapotřebí takto získané hodnoty z XML dokumentu naplnit do HTML formuláře. Můžeme jej plnit přímo jako text do HTML kódu nebo například do input boxů. Například pomocí tagu `<input type=‘‘text‘‘>` vytvoříme input box. Dále je zapotřebí pomocí tagu `<xsl:attribute name=‘‘id‘‘>` nastavit hodnotu ID daného input boxu a pomocí tagu `<xsl:attribute name=‘‘value‘‘>` nastavit hodnotu input boxu. Výsledkem převodu výše zmíněných dokumentů XML a XSL je HTML formulář, viz. obrázek č. 4.2.



Obrázek 4.2: Ukázka HTML formuláře

Dále bylo zapotřebí definovat vzhled šablony pro PDF dokument. Jak již bylo zmíněno v podkapitole č. 3.2, jedná se o XSL dokument. Ukázka XSL dokumentu pro PDF dokument je uvedena v příloze C. Nejdůležitější části XSL dokumentu pro PDF dokument jsou části, ve kterých se definuje cesta k hodnotám jednotlivých tagů XML dokumentu. Tyto části jsou stejné jako části již výše zmíněné v XSL dokumentu pro HTML formulář. Výsledkem převodu výše zmíněných dokumentů XML a XSL je PDF dokument viz. obrázek č. 4.3.



Obrázek 4.3: Ukázka PDF dokumentu

Vytváření reálných šablon pro HTML formulář a PDF dokument probíhalo obdobně jako v předchozím příkladu.
Jako ukázkové šablony byly vybrány následující:

- Protokol dárce krve.
- Uzávěrka skladu.

V příloze D je ukázka výsledného HTML formuláře protokolu dárce krve (viz. obrázek č. D.1) a výsledný PDF dokument protokolu dárce krve (viz. obrázek č. D.2). Dále je tu ukázka výsledného HTML formuláře uzávěrky skladu (viz. obrázek č. D.3) a výsledný PDF dokument uzávěrky skladu (viz. obrázek č. D.4)

Kapitola 5

Testování

Testování probíhalo za účelem ověření splnění na začátku stanovených cílů. Zároveň bylo zapotřebí otestovat správnost chování řešení na situacích, které mohou v rámci používání nastat.
Testování probíhalo z více stran:

- Testování z mé strany jakožto vývojáře.
- Testování ze strany vybrané vzorky uživatelů.

Celé testování probíhalo nad dvěma ukázkovými šablonami. Jednalo se o protokol dárce krve a uzávěrku skladu, viz. příloha D.

Body testování, na které jsem se jakožto vývojář nejvíce zaměřil:

- Stabilita řešení.
- Správnost chování.
- Ošetření různých chybových stavů.

V rámci testování stability řešení jsem se zaměřil na neustálé opakování kroků. Kroků jako je výběr šablony, nahrání šablony a samotné generování PDF dokumentu. Při testování byl zjištěn problém a to konkrétně problém opozděného vytvoření požadovaného PDF dokumentu. Proto bylo přidáno testování na existenci požadovaného PDF dokumentu. V reálné situaci to funguje tak, že se otestuje URL PDF dokumentu, pokud neexistuje, daný proces se na určenou dobu uspí a následně se provede znovu test na URL PDF dokument.

V rámci testování správnosti chování řešení byla testována funkcionalita řešení na různých webových prohlížečích.
Jednalo se o:

- Google Chrome (verze 34.0.1847.131).
- Opera (verze 20.0.1387.91).
- Mozilla Firefox (verze 28.0).
- Internet Explorer (verze 9.0.8112.16421).

V průběhu testování se zjistili menší i větší problémy. Mezi menší problémy patřili malé deformace vzhledu uživatelského rozhraní. Mezi větší problémy patřil problém nefunkčnosti generování PDF dokumentu. Tento problém byl konkrétně na webovém prohlížeči Mozilla Firefox. Všechny výše zmíněné problémy byli odstraněny podmíněním a upravením chování pro výše zmíněné webové prohlížeče. Kromě samotného testování funkcionality řešení na různých webových prohlížečích byla testována i na správnost chování šablon. Tady bylo důležité ověřit zda se všechny vyplněné položky HTML formuláře přesunou na správné místo PDF dokumentu. To znamená že se správně spárovali jednotlivé položky.

V rámci ověření ošetření různých chybových stavů jsem prováděl různé kombinace kroků. Kroky jako je požadavek o šablonu bez vybrané šablony a generování PDF dokumentu bez nahrané šablony.

Pro testování s použitím vzorky uživatelů byla použita metoda přímého sledování uživatele. U přímého sledování uživatele jsem se nejvíce zaměřil na správnost a pochopení práce s uživatelským rozhraním. Žádný z vybraných uživatelů neměl potíže s obsluhou uživatelského rozhraní. Tím byla ověřena i úspěšnost cíle vytvořit uživatelské rozhraní, které je jednoduché a přehledné. Z toho také vyplývá čas strávený jednotlivými prováděnými úkony vedoucími k požadovanému PDF dokumentu. Do času se nepočítal čas strávený vyplňováním šablony. Průměrný čas strávený vytvořením jednoho PDF dokumentu vybraným vzorkem uživatelů se pohyboval kolem 15 sekund. Při používaní starého řešení, kdy bylo zapotřebí více externích nástrojů, byl průměrný čas strávený vytvořením jednoho PDF dokumentu téměř dvojnásobný.

Kapitola 6

Závěr

Dnes, tvůrci IS na zakázku musí při potřebě tisku formulářových dat do PDF dokumentů využívat různé externí nástroje. Používání externích nástrojů přináší mnoho problémů a zbytečnou spotřebu času.

Cílem této práce bylo vytvořit škálovatelné řešení pro snadné generování PDF dokumentů z vyplněných formulářů. Samotné implementaci předcházelo studium existujících řešení, technologií z dané oblasti a návrh řešení jednotlivých částí tiskového serveru pro vytváření PDF dokumentů. Navržené řešení nabízí systém, který zefektivňuje definici formulářových dat a jejich tisk do PDF dokumentu pomocí šablon.

Výsledkem práce je funkční škálovatelné řešení, které umožňuje snadno a bez použití externích komerčních nástrojů realizovat generování HTML formulářů a tisk formulářových dat do PDF dokumentů na základě šablon. To znamená, že si uživatel může vybrat ze seznamu dostupných šablon požadovanou šablonu, vyplnit ji a převést na PDF dokument. Celá obsluha řešení probíhá přes webové uživatelské rozhraní. Obsluhu jednotlivých požadavků zabezpečuje aplikační server. Data pro šablony jsou uloženy v databázi Caché. Díky zavedenému systému verzování má uživatel vždy k dispozici aktuální verzi šablony.

Testování řešení odhalilo menší i větší problémy. Většinou se jednalo o problémy způsobené používáním různých webových prohlížečů. Následným podmíněním pro různé webové prohlížeče byli tyto problémy odstraněny.

Do budoucna je naplánováno rozšíření funkcionality o množství funkcí. Jedná se například o funkci autentizace pro přístup k službě. Díky autentizaci pak bude možné vytvořit funkce pro historii vytvořených PDF dokumentů, ukládání rozpracovaných šablon a vytváření vlastních šablon. Dále je v plánu přesunout všechny informační texty a hlášky do databáze. Tím se umožní vytvořit i vícejazyčné řešení. V neposlední řadě je v plánu vytvořit vlastní editor XSL dokumentu, popřípadě najít jednodušší cestu pro jeho vytváření.

Literatura

[1] Foundation, T. A. S.: The Apache FOP Project [online]. http://xmlgraphics.apache.org/fop/, [cit. 2014-1-8].

[2] Holoubek, M.: InterSystems Caché Post-Relational Database [online]. http://www.fi.muni.cz/~kripac/PV136/holoubek.pdf, [cit. 2012-12-08].

[3] InterSystems: Using Java with Caché [online]. http://www.odbms.org/wp-content/uploads/2014/02/Using-Java-With-Caché.pdf, 2009-06-30 [cit. 2012-12-14].

[4] InterSystems: Caché - High performance object database. http://www.intersystems.com/cache/, [cit. 2012-12-08].

[5] NetBeans: Getting Started with JAX-WS Web Services [online]. http://netbeans.org/kb/docs/websvc/jax-ws.html, [cit. 2013-1-20].

[6] TESLA_ELTOS: *MUMPS*. 19 [cit. 2012-12-08].

[7] UNIpress: Přehled verzí PDF [online]. http://www.unipress.cz/index.php/cs/parametry-pdf/verze-pdf, 2012-12-14 [cit. 2012-10-24].

[8] W3Schools: SOAP Tutorial [online]. http://www.w3schools.com/webservices/ws_soap_intro.asp, [cit. 2013-1-12].

[9] W3Schools: XSL-FO Tutorial [online]. http://www.w3schools.com/xslfo/, [cit. 2014-1-8].

[10] W3Schools: XSLT Tutorial [online]. http://www.w3schools.com/xsl/, [cit. 2014-1-8].

[11] Zendulka, J.: Objektově-relační databáze [online]. https://www.fit.vutbr.cz/study/courses/PRD/private/lectures/2_ORdatabaze_2.pdf, 2012 [cit. 2013-1-12].

[12] Zendulka, J.: Úvod, základní pojmy databázových technologií [online]. https://www.fit.vutbr.cz/study/courses/IDS/private/prednasky/1_uvod.pdf, 2012 [cit. 2013-1-12].

Dodatek A

Ukázka zápisu XML

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE root [
<!ATTLIST jmeno id ID #IMPLIED>
<!ATTLIST prijmeni id ID #IMPLIED>
<!ATTLIST email id ID #IMPLIED>
]>
<root>
  <osobni_udaje>
    <jmeno id="jmeno" caption="Jmeno:">
      <value>Test</value>
    </jmeno>
    <prijmeni id="prijmeni" caption="Prijmeni:">
      <value>Testovaci</value>
    </prijmeni>
    <email id="email" caption="Email:">
      <value>testovaci@test.cz</value>
    </email>
  </osobni_udaje>
</root>
```

Dodatek B

Ukázka zápisu XSL dokumentu pro HTML formulář

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <html>
      <body style="font-family: sans-serif">
        <form>
          <table border="0">
            <xsl:for-each select="root/osobni_udaje/*">
              <tr>
                <td>
                  <xsl:value-of select="@caption"/>
                </td>
                <td>
                  <input type="text">
                    <xsl:attribute name="id">
                      <xsl:value-of select="@id"/>
                    </xsl:attribute>
                    <xsl:attribute name="value">
                      <xsl:value-of select="value"/>
                    </xsl:attribute>
                  </input>
                </td>
              </tr>
            </xsl:for-each>
          </table>
        </form>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

Dodatek C

Ukázka zápisu části XSL dokumentu pro PDF dokument

```
<xsl:template match="root">
  <fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
    <fo:layout-master-set>
      <fo:simple-page-master master-name="simpleA4"
        page-height="29.7cm" page-width="21cm"
        margin-top="2cm" margin-bottom="2cm"
        margin-left="2cm" margin-right="2cm">
        <fo:region-body/>
      </fo:simple-page-master>
    </fo:layout-master-set>
    <fo:page-sequence master-reference="simpleA4">
      <fo:flow flow-name="xsl-region-body">
        <fo:block font-size="10pt">
          <fo:table table-layout="fixed" width="100%"
            border-collapse="separate">
            <fo:table-column column-width="4cm"/>
            <fo:table-column column-width="4cm"/>
            <fo:table-column column-width="5cm"/>
            <fo:table-body>
              <xsl:apply-templates select="osobni_udaje"/>
            </fo:table-body>
          </fo:table>
        </fo:block>
      </fo:flow>
    </fo:page-sequence>
  </fo:root>
</xsl:template>
<xsl:template match="osobni_udaje">
  <fo:table-row>
    <fo:table-cell>
      <fo:block> <xsl:value-of select="jmeno"/> </fo:block>
    </fo:table-cell> ... podobne pro prijmeni a email
```

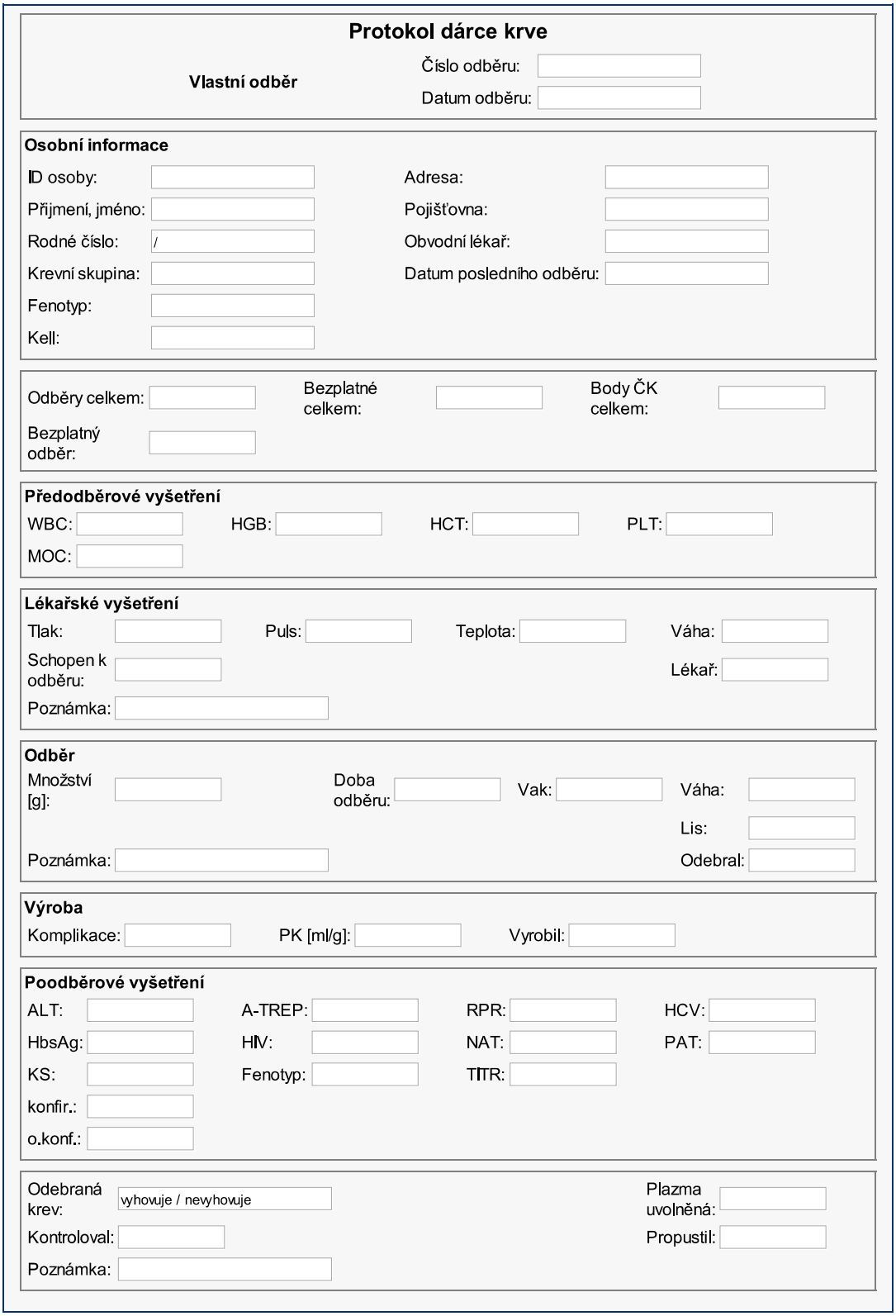
Dodatek D

Ukázkový HTML formulář a výsledný PDF dokument

Jako ukázkové šablony byly vybrány následující:

- Protokol dárce krve.
- Uzávěrka skladu.

Níže je vidět HTML formulář (viz. obrázek č. D.1) a PDF dokument (viz. obrázek č. D.2) pro protokol dárce krve. Dále je tu vidět HTML formulář (viz. obrázek č. D.3) a PDF dokument (viz. obrázek č. D.4) pro uzávěrku skladu.



Obrázek D.1: Ukázka HTML formuláře protokolu dárce krve

Protokol dárce krve

Vlastní odběr

Číslo odběru: 14010002
Číslo odběru: 31.12.2019

ID osoby: 3888
Přijmení, jméno: Smith John
Rodné číslo: 123456/5858
Krevní skupina: A
Fenotyp: -
Kell: -

Adresa: Bulharská 129
Pojišťovna: VZP-110
Obvodní lékař: Franta Mrkvička
Datum posledního odběru: 12.03.2018

Odběry celkem: 10
Bezplatný odběr: ano

Bezplatné celkem: 10

Body ČK celkem: 1500

Předodběrové vyšetření

WBC: --
HGB: --
HCT: --
PLT: --
MOC: --

Lékařské vyšetření

Tlak: 120/90
Puls: 75
Teplota: 37
Váha: 90kg
Schopen k odběru: ano
Lékař: MIK
Poznámka: schopen

Odběr

Množství [g]: 100
Doba odběru: 16:00
Vak: 2
Váha: 100g
Lis: -
Poznámka: ...
Odebral: LOP

Výroba

Komplikace: ne
PK [ml/g]: 120
Vyrobil: KUK

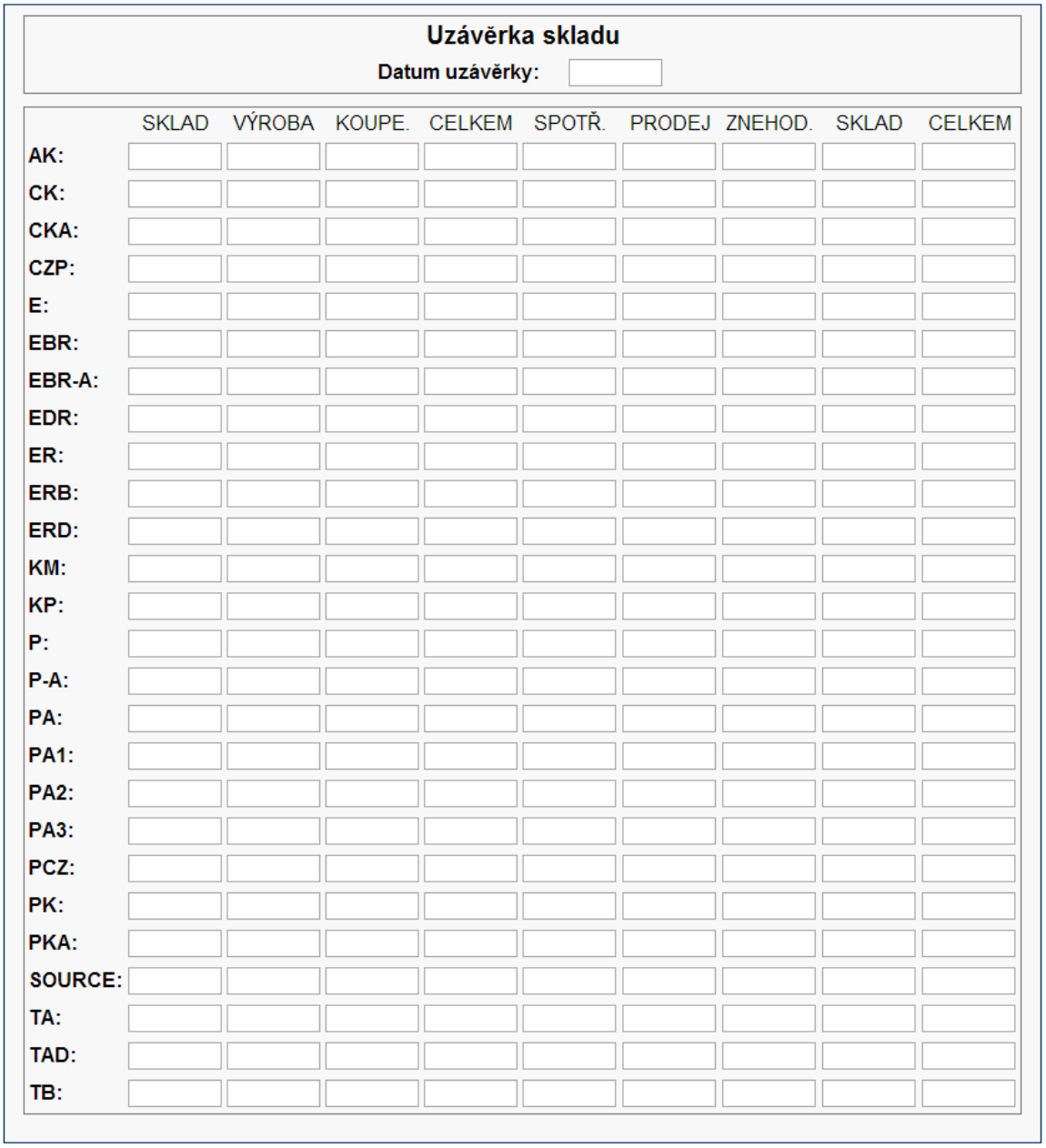
Poodběrové vyšetření

ALT: --
A-TREP: --
RPR: --
HCV: --
HbsAg: --
HIV: --
NAT: 2
PAT: --
KS: 0
Fenotyp: --
TITR: --
konfir.: --
o.konf.: --

Odebraná krev: vyhovuje
Kontroloval: PIP
Poznámka: vyhovuje

Plazma uvolněná: ano
Propustil: PIP

Obrázek D.2: Ukázka výsledného PDF dokumentu protokolu dárce krve



Obrázek D.3: Ukázka HTML formuláře uzávěrky skladu

Uzávěrka skladu

Datum uzávěrky: 31.04.15

	SKLAD	VÝROBA	KOUPE.	CELKEM	SPOTŘ.	PRODEJ	ZNEHOD.	SKLAD	CELKEM
AK:	1	2	3	4	5	6	7	8	9
CK:	10	11	12	13	14	15	16	17	18
CKA:	19	20	21	22	23	24	25	26	27
CZP:	28	29	30	31	32	33	34	35	36
E:	37	38	39	40	41	42	43	44	45
EBR:	46	47	48	49	50	51	52	53	54
EBR-A:	55	56	57	58	59	60	61	62	63
EDR:	64	65	66	67	68	69	70	71	72
ER:	73	74	75	76	77	78	79	80	81
ERB:	82	83	84	85	86	87	88	89	90
ERD:	91	92	93	94	95	96	97	98	99
KM:	101	102	103	104	105	106	107	108	109
KP:	110	111	112	113	114	115	116	117	118
P:	119	120	121	122	123	124	125	126	127
P-A:	128	129	130	131	132	133	134	135	136
PA:	137	138	139	140	141	142	143	144	145
PA1:	146	147	148	149	150	151	152	153	154
PA2:	155	156	157	158	159	160	161	162	163
PA3:	164	165	166	167	168	169	170	171	172
PCZ:	173	174	175	176	177	178	179	180	181
PK:	182	183	184	185	186	187	188	189	190
PKA:	191	192	193	194	195	196	197	198	199
SOURCE:	200	201	202	203	204	205	206	207	208
TA:	209	210	211	212	213	214	215	216	217
TAD:	218	219	220	221	222	223	224	225	226
TB:	227	228	229	230	231	232	233	234	235

Obrázek D.4: Ukázka výsledného PDF dokumentu uzávěrky skladu

Dodatek E

Obsah CD

Přiložené CD obsahuje:

- Zdrojové soubory v LaTeXu k technické zprávě (adresář TZ/src).
- Technickou zprávu ve formátu PDF (adresář TZ).
- Plakát z pohledu technologií (adresář Plakat).
- Plakát z pohledu řešení (adresář Plakat).
- Demonstrační video (adresář Video).
- Zdrojové soubory řešení a projekt pro Netbeans (adresář NetBeans-project).
- Instalační soubory pro výsledné řešení (adresář Instalace).
- Ukázkové šablony pro protokol dárce krve a uzávěrku skladu (adresář Instalace/Sablony).

Dodatek F

Instalační manuál

Tato příloha se bude zabývat jednotlivými kroky vedoucími k úspěšné instalaci celého řešení. Jak v případě klienta tak serveru se předpokládá s existencí nainstalovaného webového serveru. Dále se předpokládá existence Caché serveru. Veškeré zmíněné soubory jsou přiloženy na CD v adresáři Instalace.
Instalace a konfigurace serveru:

- Instalace přiloženého souboru Print_Server-Server.war na webový server.
- Do adresáře config dané domény vložit přiložený konfigurační soubor GlassFish/-fop.xconf (konfigurace pro FOP, který se po užívá pro vytváření PDF dokumentů).
- Do adresáře config dané domény vložit přiložený konfigurační soubor GlassFish/Print-Server.xml.
- Úprava PrintServer.xml

Příklad a popis konfiguračního souboru PrintServer.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<config>
  <cache><!--Nastaveni pro databazi Cache-->
          <host>localhost</host>
          <username>_SYSTEM</username>
          <password>SYS</password>
          <port>1972</port>
          <nameSpace>SAMPLES</nameSpace>
  </cache>
  <server><!--Nastaveni pro server obsahujici webove sluzby-->
    <!--Cesta adresare pro praci se sablonami
        Je dulezite aby tato cesta byla pristupna pro
        GlassFish server. Nejlepe je proto tuto cestu
        nastavit do instalacniho adresare souboru war. -->
          <context_root>...\\templates\\</context_root>
    <!--Vlastni URL adresa-->
          <url>http://192.168.1.1:8080/Print_Server-Server/</url>
  </server>
</config>
```

Instalace a konfigurace klienta:

- Instalace přiloženého souboru Print_Server-Klient.war na webový server.
- Vstup do instalačního adresáře a následně do adresáře wsdl.
- Úprava IP adresy a portu v souboru GeneratePDF.wsdl.
- Úprava IP adresy a portu v souboru LoadTemplate.wsdl.

Úprava IP adresy a portu v souboru WSDL spočívá v úpravě tagů:

- `<xsd:import schemaLocation>`
- `<soap:address location>`

Instalace tříd a rutin do Caché:

- Spuštění management portálu.
- Importovat a přeložit třídy z přiloženého souboru Caché/class.xml.
- Importovat a přeložit rutiny z přiloženého souboru Caché/routines.ro.

Inicializace struktury dat pro uložení šablon v Caché:

- Spuštění terminálu Caché.
- Zvolení příslušného namespace.
- Zadání příkazu `d CreateTreeTemplates^Templates`.

Nahrávání ukázkových šablon:

- Spuštění terminálu Caché.
- Zvolení příslušného namespace.
- Zadání příkazu `d FillTemplate^Templates(templateName,version)`.
- Spuštění management portálu.
- Naplnit nově vytvořené struktury, viz. podkapitola č. 3.5, obrázky č. 3.4 a 3.5, obsahem souborů tvořících šablonu, viz. podkapitola č. 3.2. Ukázkové šablony se nacházení v adresáři Sablony.

Dodatek G

Plakát

Obrázek G.1: Plakát